UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación
(EUR-ACE®)

Reconocimiento de escenas mediante integración multiescala de redes convolucionales.

Autor: Pablo Collado Recio.
Tutor: Miguel Ángel García García.

TRABAJO FIN DE GRADO

Junio 2021

Reconocimiento de escenas mediante integración multiescala de redes convolucionales.

Autor: Pablo Collado Recio.

Tutor: Miguel Ángel García García.

Video Processing and Understanding Lab
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio 2021

Trabajo parcialmente financiado por el Ministerio de Economía y Competitividad del Gobierno de España bajo el proyecto TEC2017-88169-R (MobiNetVideo) (2018-2020)

Resumen

En el siguiente Trabajo de Fin de Grado se lleva a cabo el entrenamiento y validación experimental de redes neuronales profundas pre-entrenadas, mediante un conjunto de datos formado por diferentes escenas de la Universidad Autonóma de Madrid. Esto tiene lugar tras haber realizado una integración multi-escala de estos conjuntos de datos como destino de diferentes tipos de redes neuronales. Para ello se crean diferentes datasets, a cada cual se le aplica una estrategia de escala diferente en función de las redes neuronales a las que estén destinadas, tanto para especializadas en escenas como en objetos. Las redes constarán de una arquitectura ResNet.

De esta forma forma se abordará la premisa señalada en *Scene recognition with CNNs: objects, scales and dataset bias*[1] y se estudiará la eficacia de la multiescala en problemas de reconocimiento de escenas. Por último, se hace un estudio comparativo del nivel de reconocimiento de estas redes, tanto independientemente como de manera integrada.

Palabras clave

Red neuronal, entrenamiento, validación, extracción de características, finetuning, escalas, epoch, arquitectura convolucional, dataset

Abstract

In the following Thesis, the training and experimental validation of deep neural networks already pre-trained is carried out, using a set of data formed by different scenes from the Autonomous University of Madrid. This takes place after having performed a multi-scale integration of these data sets as a destination for different types of neural networks. For this, different datasets are created, each of which is applied a different scale strategy depending on the neural networks to which they are intended, both for those specialized in scenes and objects. The networks will consist of a ResNet architecture.

This way, the premise indicated in *Scene recognition with CNNs: objects, scales and dataset bias*[1] will be addressed and the effectiveness of multiscale in scene recognition problems will be studied. Finally, a comparative study is made of the level of recognition of these networks, both independently and in an integrated manner.

Keywords

Neural network, training, validation, feature extraction, finetuning, scales, epoch, convolutional architecture, dataset

Agradecimientos

Me gustaría agradecer primero de todo a mis campañeros que me han acompañado durante esta etapa tan crucial de la vida. También a todos los profesores que me han impartido clase, en especial a Miguel Ángel, mi tutor, por su paciencia. Y por último pero no por ello menos importante a mi familia y pareja. Tardó más de lo esperado, pero se consiguió.

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Motivación

El estado del arte en reconocimiento visual se basa en la combinación exitosa de representaciones de imágenes y conjuntos de datos masivos. Las redes neuronales convolucionales (CNN en adelante) entrenadas en ImageNet, logran un alto rendimiento en el reconocimiento de objetos, mientras que las CNN entrenadas con Places365 lo obtienen en el reconocimiento de escenas. Aunque, las CNNs también tienen limitaciones, como la falta de invariancia a una escala significativa. Este problema es particularmente importante en el reconocimiento de escenas, debido a una gama más amplia de escalas y a una mayor cantidad de objetos por imagen cuando éstas son muy grandes.

En general, se suele utilizar las entrenadas en ImageNet para extraer las características particulares en lugar de las de Places, ya que las características locales están más próximas a los objetos que a las escenas. Sin embargo, un aspecto pasado por alto en este escenario de múltiples escalas es el papel de la misma y su relación con la extracción de caracteristicas (feature-extractor). Un obstáculo de los enfoques multiescala actuales es el uso ingenuo de CNNs, simplemente considerando éstas como extractores de características de propósito general. Utilizar el mismo modelo fijo de CNN para todas las escalas conduce inevitablemente al sesgo del conjunto de datos, ya que las propiedades de los datos varían a diferentes escalas, mientras que el extractor de características permanece fijo. En este caso, este sesgo en la distribución de características se induce al escalar la imagen. Si la operación de escalado es considerable, las características de los datos pueden cambiar por completo, pasando de datos de escena a datos de objetos. Comprender y cuantificar este sesgo puede ayudarnos a diseñar mejores arquitecturas multiescala híbridas e incluso mejores formas

de combinar el conocimiento sobre objetos y escenas.

Esta memoria de Trabajo de Fin de Grado es la síntesis del análisis basado en este tipo de redes neuronales, sobre el cual se agrupan los diferentes conocimientos adquiridos desde cero, tras una investigación sobre esta materia, las pruebas realizadas sobre ellas y la exposición de los resultados obtenidos.

1.2. Objetivos

Como se propone en el artículo[1] este trabajo tiene como objetivo reproducir la técnica definida, basada en la integración multiescala de redes neuronales profundas pre-entrenadas con conjuntos de datos centrados en objetos (ImageNet), escenas (Places365) y su posterior evaluación a partir de una base de datos de imágenes de escenas del campus de la Universidad Autónoma. Para ello se creará, a partir del dataset original obtenido tras la captura de imágenes, diferentes conjuntos de datos que consistentes en pequeñas modificaciones en el tamaño de estas y la selección de diferentes muestras. Las redes neuronales con las se trabajará serán las ResNet50, previamente entrenadas para Places e Imagenet.

Los objetivos propuestos para este Trabajo de fin de grado son los siguientes:

• Comprender el funcionamiento de una red neuronal convolucional, así como su arquitectura interna.

• Demostar la limitación de uso al aplicar de forma independiente diferentes CNNs como extractor de características, debido al sesgo del conjunto de datos inducido por cambios de escala.

• Implantar una estrategia para mitigar el sesgo perteneciente al conjunto de datos mediante la integración de redes y escalas específicas: híbrido entre las redes pre-entrenadas con bases de datos de escenas y objetos con una arquitectura ResNet.

• Analizar el grado de acierto de la red integrada en función con respecto a las diferentes clases personalizadas.

1.3. Estructura de la memoria

La memoria consta de los siguientes capítulos:

- Capítulo 1 Introducción.
- Capítulo 2 Estado del arte.

- Capítulo 3 Diseño.
- Capítulo 4 Desarrollo.
- Capítulo 5 Pruebas y resultados finales.
- Capítulo 6 Conclusiones y trabajo futuro.

Capítulo 2

Estado del arte

2.1. Introducción

En este capítulo se describirá una visión general de la tecnología ya existente, que guarda conexión con este proyecto y que se ha utilizado para llevar a cabo éste. Por lo tanto se expondrán los elementos principales que fundamentan el trabajo realizado en esta tarea, como son las redes neuronales y los motores de procesamiento de éstas, así como los diferentes entornos de trabajo y sus lenguajes correspondientes, haciendo un análisis técnico y fundamental para el conocimiento de estas tecnologías.

2.2. Python

Es un lenguaje de programación interpretado, orientado a objetos y de alto nivel con semántica dinámica. Sus estructuras de datos integradas de alto nivel, combinadas con tipado y enlace dinámico, lo hacen muy atractivo para el desarrollo rápido de aplicaciones, así como para su uso como lenguaje de scripts para conectar componentes existentes. La sintaxis simple y fácil de aprender de Python enfatiza en la legibilidad y, por lo tanto, reduce el coste de mantenimiento del programa. Python admite módulos y paquetes, lo que fomenta la modularidad del programa y la reutilización del código. El intérprete de Python y su extensa biblioteca están disponibles en formato fuente o binario de forma gratuita para todas las plataformas principales, y permite su libre distribución. Se ha utilizado este lenguaje para la realización de este estudio debido a su fácil uso, comprensión y el gran número de paquetes y librerías (entre las que se encuentra Pytorch). Destaca por su versatilidad y flexibilidad, y es de gran utilidad para este trabajo porque comprende una gran variedad de herramientas para trabajar en campos englobados dentro de la Inteligencia Artificial.

2.3. GPU

La unidad de procesamiento de gráficos, o GPU, se ha convertido en uno de los tipos más importantes de tecnología, tanto para la informática personal como empresarial. Diseñada para el procesamiento en paralelo, la GPU se utiliza en una amplia gama de aplicaciones, incluida la reproducción de gráficos y vídeo. Aunque son más conocidas por sus capacidades en juegos, las GPU se están volviendo más populares para su uso en producción creativa e Inteligencia Artificial.

Las GPUs se diseñaron originalmente para acelerar la renderización de gráficos 3D. Con el tiempo, se volvieron más flexibles y programables, mejorando sus capacidades. Esto permitió a los programadores gráficos crear efectos visuales más interesantes y escenas realistas con técnicas avanzadas de iluminación y sombreado. Otros desarrolladores también comenzaron a aprovechar el poder de las GPU para acelerar drásticamente las cargas de trabajo adicionales en computación de alto rendimiento (HPC), aprendizaje profundo y más.

Algunas de las aplicaciones más interesantes para la tecnología GPU incluyen la inteligencia artificial y el aprendizaje automático. Debido a que incorporan una cantidad extraordinaria de capacidad computacional, pueden ofrecer una aceleración muy potente en las cargas de trabajo, ya que aprovechan la naturaleza altamente paralela de las GPU, como el reconocimiento de imágenes. Muchas de las tecnologías de aprendizaje profundo actuales dependen de que las GPUs funcionen junto con las CPUs.

La GPU empleada para la realización de este trabajo ha sido la GTX 1050Ti de NVIDIA con 4 GB de RAM.

2.4. CUDA

Es una plataforma de computación paralela[2] y un modelo de programación desarrollado por NVIDIA para la computación general en unidades de procesamiento gráfico (GPU). Con ella, los desarrolladores pueden acelerar drásticamente las aplicaciones informáticas aprovechando la potencia de las GPU. Son las siglas de Compute Unified Device Architecture (CUDA) que hace referencia a una plataforma de computación en paralelo incluyendo un compilador y un conjunto de herramientas de desarrollo creadas por NVIDIA que permiten a los programadores usar una variación del lenguaje de programación C/C++ para codificar algoritmos en GPU de NVIDIA.

Esta arquitectura intenta explotar las ventajas de las GPU frente a las CPU de propósito general utilizando el paralelismo que ofrecen sus múltiples núcleos, que

permiten el lanzamiento de un altísimo número de hilos simultáneos. Simplificando al máximo, un núcleo CUDA (CUDA Core) equivale a un mini procesador que se encarga de cierto tipo de instrucciones, que suelen poder ejecutarse de manera paralela. La tarjeta empleada (GTX 1050Ti) posee 768 CUDA Cores.

2.5. Redes neuronales convolucionales

Una red neuronal convolucional[1] (CNN) es una clase de red neuronal especializada en procesar datos que tienen una topología similar a una cuadrícula, como una imagen; siendo ésta una representación binaria de datos visuales que contiene una serie de píxeles dispuestos en forma de cuadrícula. Este tipo de red[3] consiste en un algoritmo de aprendizaje profundo que puede tomar una imagen de entrada, asignarle importancia (pesos y sesgos aprendibles) a varios aspectos y objetos de la misma, y así poder diferenciarlas. Esta clase de red es la más utilizado con respecto a todas las tareas de visión artificial, como son la clasificación y división o segmentación de imágenes que se han utilizado en este proyecto de fin de grado. Se modelan como colecciones de neuronas que están conectadas entre sí, cada una de éstas recibe una entrada, realiza una serie de operaciones y opcionalmente lo conecta de forma no lineal con otra neurona distinta. Desde su concepción, las CNNs[4] se han caracterizado por sus conexiones locales, reparto de pesos y agrupación local. Las dos primeras propiedades permiten que el modelo descubra patrones visuales informativos locales con menos parámetros ajustables. La última propiedad equipa a la red con invarianza de transferencia. El excelente desempeño de este tipo de redes se puede atribuir en gran medida a estas propiedades; así como ciertas estructuras con pesos aleatorios, también podrían lograr unos buenos resultados.

La arquitectura 2.1 de una CNN está conformada tres capas: una capa convolucional, una capa de pooling (agrupación) y una capa fully-connected.

- Capa convolucional: realiza un producto escalar entre dos matrices, donde una matriz se define como el conjunto de parámetros que se pueden aprender, también conocida como kernel, y la otra es la porción restringida del campo receptivo (imagen de entrada). El kernel es espacialmente más pequeño que una imagen, pero tiene más profundidad.

- Capa de pooling: reemplaza la salida de la red en ciertas ubicaciones, generando una estadística resumida de las salidas cercanas. Esto ayuda a reducir el tamaño espacial de la representación, lo que disminuye la cantidad requerida de cálculos y pesos. La operación de agrupación se procesa en cada segmento de la representación de forma individual. Hay varios tipos de agrupación, como el promedio (average

pooling), que es una media ponderada basada en la distancia desde el píxel central y la normalización de sus vecinos. Sin embargo, el proceso más estandarizado es la agrupación máxima (max pooling), que informa la salida máxima del vecindario.

- Capa fully-connected: las neuronas de esta capa tienen conectividad total con todas las neuronas de la capa anterior y posterior. Ayuda a mapear la representación entre la entrada y la salida.

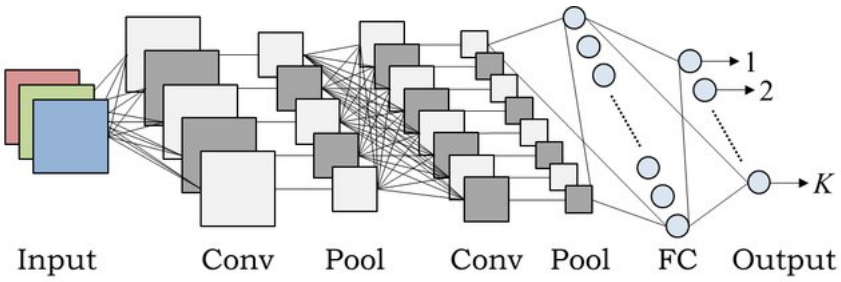

Figura 2.1: Arquitectura de Red Neuronal Convolucional[5]

2.6. Frameworks de Deep Learning

Los frameworks de Deep Learning son entornos de trabajo que ofrecen bloques de construcción para diseñar, entrenar y validar redes neuronales profundas, a través de programación de alto nivel. Constan de una interfaz, biblioteca o herramienta que permite construir modelos de aprendizaje profundo de manera más fácil y rápida, sin entrar en los detalles de los algoritmos y arquitectura internos. Proporcionan una forma clara y concisa de definir modelos utilizando una colección de componentes prediseñados y optimizados.

2.6.1. Pytorch

PyTorch[6] es un paquete informático científico basado en Python que utiliza el poder de las unidades de procesamiento de gráficos (GPUs). Es también una de las plataformas de investigación de aprendizaje profundo preferidas, construida para proporcionar la máxima flexibilidad y velocidad. Conocida por proporcionar dos de las funciones de más alto nivel: el cálculo de tensores con un fuerte soporte de aceleración de GPU y la construcción de redes neuronales profundas.

Realiza la ejecución inmediata de cálculos tensoriales dinámicos con diferenciación automática y aceleración de la GPU, y lo hace manteniendo un rendimiento comparable al más rápido de las bibliotecas para el aprendizaje profundo. Consta de numerosas librerías de Python que tienen el potencial de cambiar la forma en que se realizan el aprendizaje profundo y la inteligencia artificial. Una de las razones clave del éxito de PyTorch es que se puede se desarrollar mediante Python y se pueden construir modelos de redes neuronales sin esfuerzo.

2.6.2. TensorFlow

Creada por el equipo de Google Brain, TensorFlow[7] es una librería de código abierto para el cálculo numérico y el aprendizaje automático a gran escala. Agrupa una gran cantidad de modelos y algoritmos de aprendizaje automático y profundo y los hace útiles mediante una metáfora común. Utiliza Python para proporcionar una API frontal para crear aplicaciones en el framework, mientras ejecuta esas aplicaciones en C ++ de alto rendimiento.

Permite a los desarrolladores crear gráficos de flujo de datos: estructuras que describen cómo se mueven los datos a través de éstos o una serie de nodos de procesamiento. Cada nodo en el gráfico representa una operación matemática, y cada conexión o borde entre nodos es una matriz de datos multidimensional, también conocida como tensor. Proporciona todo esto para el programador a través del lenguaje Python. Los nodos y tensores en TensorFlow son objetos de Python, y las aplicaciones de TensorFlow son en sí mismas aplicaciones de Python.

Sin embargo, las operaciones matemáticas reales no se realizan en Python. Las librerías de transformaciones que están disponibles a través de TensorFlow son de C ++. Python simplemente dirige el tráfico entre las piezas y proporciona abstracciones de programación de alto nivel para unirlas.

2.6.3. Caffe

Creado pensando en la expresión, la velocidad y la modularidad. Está desarrollado por Berkeley AI Research (BAIR). Optimizado para velocidad, modularidad y escalabilidad, este framework proporciona soluciones tanto para proyectos de investigación académica como para aplicaciones industriales en inteligencia artificial. Su especialización se basa en lenguaje y visión artificial y multimedia.

Las características clave de Caffe[8] incluyen soporte para Unidades de Procesamiento Central (CPUs) y Unidades de Procesamiento de Gráficos (GPUs), así como Compute Unified Device Architecture (CUDA) de NVIDIA y la librería cuDNN (CU-

DA Deep Neural Network). Por lo tanto, es un framework diseñado principalmente para la velocidad. Proporciona a los científicos y profesionales multimedia un framework limpio y modificable para el desarrollo de algoritmos de aprendizaje profundo state-of-the-art y una colección de modelos de referencia. Permite la experimentación y el cambio perfecto entre plataformas para la facilidad de desarrollo e implementación desde máquinas de creación de prototipos hasta entornos cloud.

2.6.4. Keras

Keras[9] es una API de aprendizaje profundo escrita en Python, que se ejecuta sobre la plataforma de aprendizaje automático TensorFlow. Fue desarrollada con un enfoque que permitiera una experimentación rápida y para ser fácil de usar, con un esquema modular. Esta API fue ”diseñada para seres humanos, no para máquinas” y ”sigue las mejores prácticas para reducir la carga cognitiva”.

Permite que las capas de la redes neuronales, las funciones de coste, los optimizadores, los esquemas de inicialización, las funciones de activación y los esquemas de regularización sean todos módulos independientes para poder combinarlos y poder crear nuevos modelos. Estos nuevos módulos son fáciles de agregar, como nuevas clases y funciones. Los modelos se definen en código Python, no en archivos de configuración de modelos separados. Permite a los ingenieros e investigadores aprovechar al máximo la escalabilidad y las capacidades multiplataforma de TensorFlow. Se puede ejecutar Keras en TPU o en grandes grupos de GPU, y puede exportar sus modelos de Keras para ejecutarlos en el navegador o en un dispositivo móvil.

Capítulo 3

Diseño

3.1. Introducción

Para afrontar la realización de este proyecto, se deben establecer una serie de premisas claras sobre cómo llevarlo a cabo y definir las bases del mismo.

El objetivo final del estudio es el reconocimiento de escenas multiescala con redes convolucionales. Se debe determinar el tipo de red neuronal convolucional que más se adecúe para este tipo de imágenes. A continuación, se procede a la elección de los pesos de los datasets con los que han sido entrenados, en función de la escala con la que lo queremos realizar. Finalmente, se expondrán las diferentes configuraciones y tecnologías que se han utilizado para cumplir los objetivos marcados.

3.2. Diseño redes neuronales

Dentro del mundo de las redes neuronales convolucionales existe una amplia gama de arquitecturas. Para la realización de este trabajo no se llevará a cabo la implementación de una red convolucional desde cero, sino que se empleará una ya existente con su arquitectura predefinida. Inicialmente se ha decido la utilización, para este proyecto, de la red convolucional ResNet50. A través de Pytorch se podrá importar y modificar algunas de las etapas internas si se considera oportuno.

3.3. Resnet50

ResNet[10] es una red neuronal convolucional profunda, dedicada a la codificación automática y clasificación de imágenes. La red elegida es ResNet50, perteneciente a la familia ResNet, conformada por 50 capas internas. Originalmente fue diseñada para

ser entrenada con el dataset de Imagenet para clasificar imágenes en 1000 categorías de objetos. El tamaño de imagen que soporta a la entrada es de 224x224 formato RGB.

Su arquitectura interna consiste en:

- Una primera convolución con un kernel de 7*7 y un max pooling de salto 2.
- 49 capas convolucionales de kernel 1*1 y 3*3.
- Un average pooling de 7*7 y una capa fully-connected, cuya salida arroja un vector que contiene los porcentajes de cada una de las 1000 categorías.

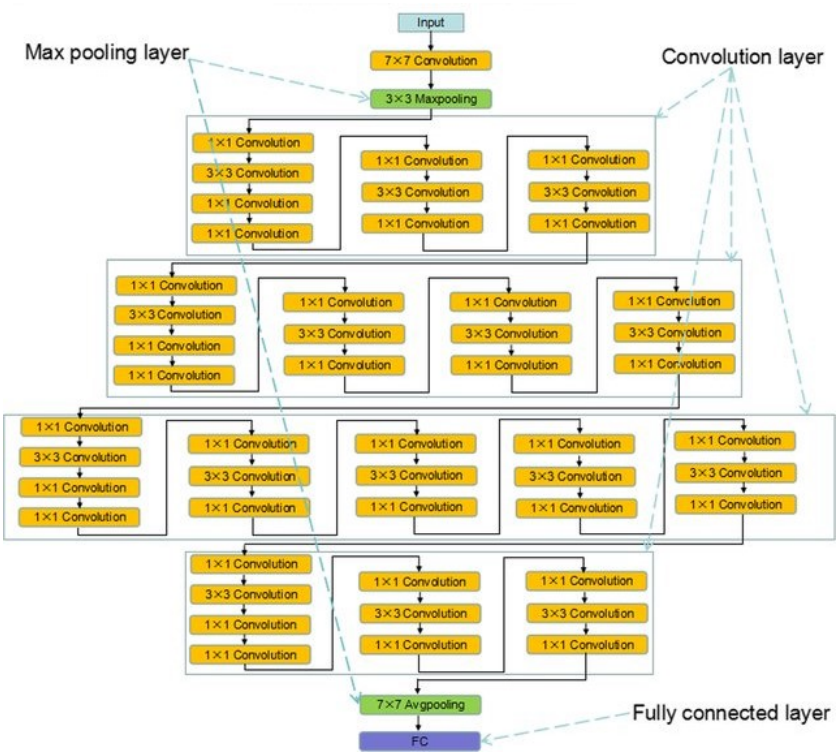

Figura 3.1: Arquitectura de ResNet50[11]

3.4. ImageNet

ImageNet[12] es una gran base de datos o conjunto de datos, con más de 14 millones de imágenes. Fue diseñado por académicos destinados a la investigación de

la visión por ordenador (Computer Vision). Fue el primero de su tipo en términos de escala. Las imágenes están organizadas y etiquetadas de forma jerárquica. Este proyecto surgió de las importantes necesidades en la investigación en el campo de Computer Vision. De esta forma, la gran cantidad de imágenes que nos proporciona ImageNet y que han sido utilizadas para pre-entrenar a la red neuronal que tomamos como base en nuestro trabajo, aportarán mucho conocimiento al objetivo final. Esta gran magnitud de imágenes; y por ende de diferentes escalas, ya sean objetos pequeños o escenas más amplias, connotará una especial importancia en el aprendizaje de la red.

3.5. PLACES365

El conjunto de datos de Places[13] está diseñado siguiendo los principios de la cognición visual humana. Su objetivo es construir un núcleo de conocimiento visual, que pueda usarse para entrenar sistemas artificiales con tareas de comprensión visual de alto nivel, como el contexto de la escena, el reconocimiento de objetos, la predicción de acciones y eventos, y la inferencia de la teoría de la mente. Las categorías semánticas de Places se definen por su función: las etiquetas representan el nivel de entrada de un entorno. En particular, consta de 1.803.460 imágenes de muestra para ser utilizadas en el entrenamiento de redes neuronales convolucionales. Su característica principal es su gran volumen de imágenes de escenas de gran escala, esenciales para el buen desempeño del sistema de reconocimiento. Fue implementada para ser utilizada en el entorno de Caffe, pero a través de PyTorch es posible importarla y adecuarla para el usode este estudio.

3.6. Fine Tuning

Un término importante a la hora de trabajar con datasets customizados es el conocido como fine tuning[14]. Éste permite la capacidad de modificar la arquitectura interna de la red. En el caso de este proyecto, se derrollará la reestructuración de la capa final (fully-connected), donde se devuelven las predicciones de la etiqueta de clase. Se llevará a cabo lo anterior para tener la misma cantidad de salidas que clases en el conjunto de datos personalizado, tanto en entrenamiento como en validación. La aplicación del fine tuning nos permite utilizar redes previamente entrenadas para reconocer las clases en las que no fueron entrenados originalmente.

3.7. Diseño de red neuronal integrada

Una vez se han determinado los diferentes tipos de redes neuronales que se van a utilizar para la realización de este trabajo, se llevará a cabo la integración de éstas para su adecuación al obejtivo a conseguir. Cada una de las redes está enfocada a un tipo de imagen con diferente escala, en la implementación del sistema de reconocimiento se requiere una única salida; fruto de la integración de las diversas redes que lo componen. Esta salida ha de ser un vector de probabilidades de tamaño equivalente al número de clases, con las que se realiza tanto el entrenamiento como la posterior validación.

La integración consiste en enlazar las últimas capas de cada una de las redes hacia una única salida, que sea del tamaño igual al número de clases personalizadas. Lo primero es adaptar cada una de las salidas, mediante la modificación de sus capas totalmente conectadas (fully-connected en inglés), las cuales componen las últimas etapas de las redes con arquitectura ResNet. Para el caso con arquitectura ResNet50, las capas fully-connected [A.10]reciben un tamaño de 2048 de la penúltima capa, la de average pool [A.11] y pasan de tener esa magnitud, a una acorde al número de clases con las que han sido pre-entrenadas (1000 para Imagenet y 365 para Places365). Esto se consigue mediante la sucesión de dos capas fully-connected como muestran las arquitecturas internas de estos dos tipos de redes en las figuras A.2 y A.1.

En este trabajo se quiere realizar el reconocimiento de escenas de la UAM, un número de clases inferior en comparativa con las clases pre-entrenadas. Por ello, a través del fine tuning se ha diseñado lo siguiente: ir disminuyendo progresivamente el tamaño de la última capa de cada red (capa fully-connected) hasta obtener un vector con un tamaño equivalente a las clases del recomendador, como se muestra en código en las figuras A.4 y A.3. De esta forma se obtiene a la salida de cada red, un tamaño correspondiente al número de clases del dataset propio de imágenes de la Universidad. Como se puede observar, la arquitectura de las redes ha sido adaptada a la necesidad de este estudio. El resultado de lo detallado anteriormente se visualiza en la figura A.5 para Places y A.6 para Imagenet.

Por último, se ha realizado una integración de las diferentes salidas, mediante la concatenación de éstas y se ha aplicado una función ReLU como se observa en código en la figura A.7. Esta función realiza una rectificación de unidad lineal, consistente en una conversión de todos los valores menores que cero en ese tensor, a cero. De esta forma se ha conseguido la integración multiescala requerida para el desarrollo del proyecto.

Capítulo 4

Desarrollo

4.1. Introducción

En este capítulo se van a explicar los cambios para la generación de las diferentes redes neuronales que constituirán el sistema de reconocimiento de las escenas. Además, se describirá la generación y estructura del dataset personalizado, así como las diferentes escalas de éste, que alimentarán tanto el entrenamiento como la validación de este proyecto.

4.2. Selección del dataset propio

El objetivo principal de este trabajo es el reconocimiento de escenas e imágenes multiescala, para ello, se ha recopilado un dataset propio de escenas de la Universidad Autónoma de Madrid. Éste está formado por una base de datos de 1000 imágenes divididas en 8 clases diferentes, que más adelante serán enumeradas. Se han tomado una serie de fotografías desde distintas perspectivas y en diferentes franjas horarias, de planos exteriores de 8 emplazamientos de la UAM, lo cual proporcionará una mayor variedad a la hora de entrenar la red neuronal, mejorando así su preparación. Con esta base de datos, se ha de desarrollar la conformación de las diferentes escalas que han de usarse para el reconocimiento de las distintas escenas. Se requieren varios tipos, con tamaño y muestras variadas, para estudiar el efecto de la multiescala en el reconocimiento de escenas.

Este dataset a su vez se subdivide en dos, que son:

• Entrenamiento: posee un tamaño aproximado de 900 imágenes, 110 para cada una de las 8 clases. Éste ha de ser de gran tamaño debido a la necesidad de una mayor diversidad de imágenes, que a la hora de entrenar la red neuronal conllevará

una mejor preaparación de ésta.

• Validación: con una extensión cercana a las 150 imágenes, 15 para cada una de las 8 clases. Con él se llevará a cabo el testeo del reconocimiento de las escenas por parte de la red. Es de extrema importancia que las imágenes de validación no sean exactas a las de entrenamiento, ni muestras de ella, para un correcto resultado a la hora de validar los aciertos.

Figura 4.1: Imagen original de muestra del dataset

4.3. Dataset para Places365

Partiendo del dataset propio que se ha generado, el cual contiene imágenes de espacios de la Universidad. Sabiendo la característica principal de las redes preentrenadas con Places365, se entrenará esta red con imágenes de escenas de gran escala y que engloben una gran multitud de objetos en ella. Se desea estudiar el efecto de la multiescala para este tipo de redes, con este fin se crean dos datasets para este tipo de imágenes. Éstas permitirán analizar como afecta el uso de diferentes escalas para redes entrenadas en Places365.

4.3.1. Escalas para Places365

Para la realización de este proyecto, se han generado dos tipos de escalas para entrenar a las redes con alto conocimiento de escenas de gran medida. El dataset propio ya posee este tipo de escala en las imágenes, con lo cual sabiendo el tamaño

que deben tener éstas a la hora de introducirlas en la red, se han creado las siguientes escalas:

• Escala gruesa superior (GS en adelante): dataset conformado con las imágenes originales tras sufrir un reescalado a 224x224 píxeles.

• Escala gruesa inferior (GI en adelante): para ésta, las imágenes se han de reescalar a un tamaño mayor al número de píxeles de entrada que espera Places, y tomar un pequeño número de muestras aleatorias, con tamaño 224x224.

De esta forma se tienen dos dataset constituidos por imágenes de gran escala que aportarán un gran conocimiento a la hora del reconocimiento de escenas. Más adelante se llevará a cabo el estudio de la elección de los parámetros de la escala GI.

Figura 4.2: Imagen de muestra del dataset de escala GS

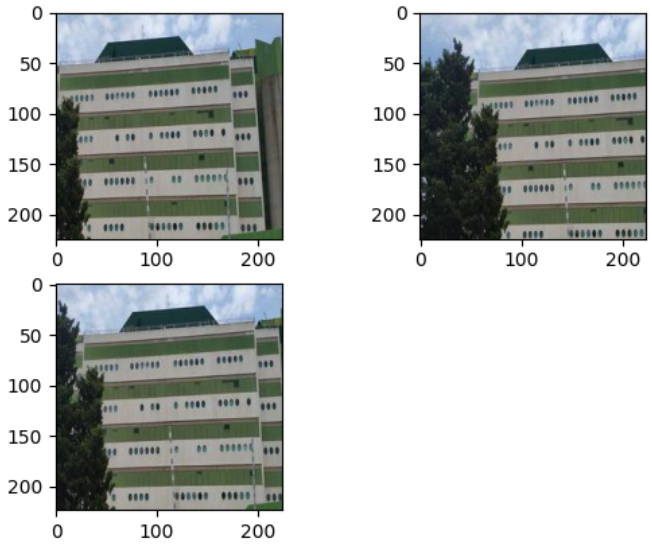

Figura 4.3: Imágenes de muestra del dataset de escala GI

4.4. Dataset para ImageNet

Al igual en el punto anteriormente mencionado, las redes entrenadas con Imagenet poseen gran conocimiento de imágenes de pequeña escala (objetos), con lo cual, cada escena posee una cantidad determinada de objetos en ella. Con este fin, se generan dos datasets para este tipo de imágenes.

4.4.1. Escalas para Imagenet

Por otro lado, partiendo del dataset original, se ha determinado la creación de dos conjuntos de datos para entrenar las redes con alto conocimiento de objetos de pequeña escala. Ya que el dataset propio no posee este tipo de escala en las imágenes originales, se ha realizado un pequeño proceso en Python para la creación de las siguientes escalas:

• Escala fina superior (a partir de ahora FS): este conjunto de imágenes han sido reescaladas a un mayor porcentaje de su tamaño original, y se han tomado una gran cantidad de muestras aleatorias de tamaño 224x224 píxeles.

• Escala fina inferior (a partir de ahora FI): este conjunto no sufre reescalado, pero se obtiene un elevado número de muestras aleatorias de cada imagen de tamaño

224x224 píxeles.

Como resultado se cuenta con dos datasets de un tamaño bastante mayor a los dos anteriores, constituidos ahora por imágenes de pequeña escala que aportarán un gran conocimiento en la fase del reconocimiento de objetos de pequeña escala. Más adelante se llevará a cabo el estudio y la elección de los parámetros, tanto de la escala FS, como FI.

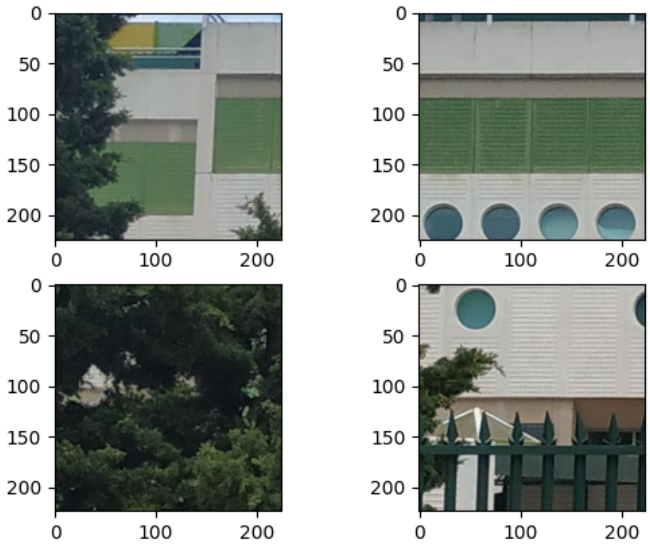

Figura 4.4: Imágenes de muestra del dataset de escala FS

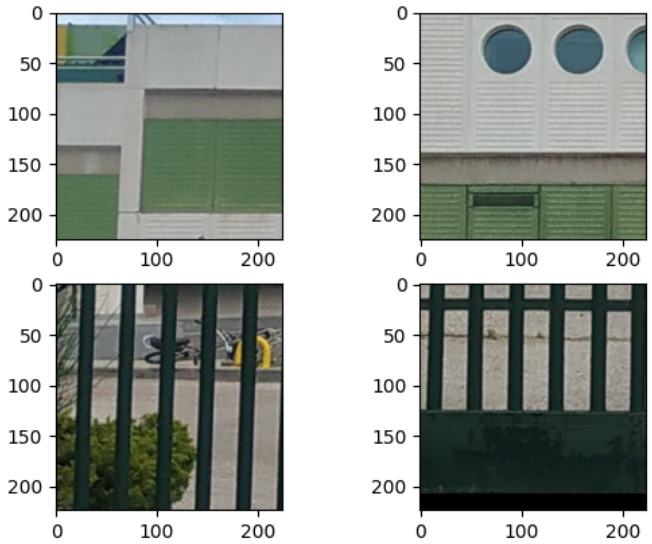

Figura 4.5: Imágenes de muestra del dataset de escala FI

Nota: la creación de las 4 escalas se lleva a cabo tanto para el dataset de entrenamiento, como para el de validación.

4.5. Modificación red neuronal

Una vez se ha difinido el número de clases que estarán bajo estudio, se procede al desarrollo de la integración de las 4 redes utilizadas. El proceso de integración, detallado en la Sección 3.7: Diseño de red neuronal integrada, seefectua mediante diferentes funciones que posee Pytorch. Para las dos redes de Imagenet, se disminuye el tamaño de la última capa (fully-conected), de tamaño 1000, progresivamente hasta obtener una capa fully-connected a la salida, de tamaño 8 [Figura A.4]. Se realiza lo mismo para las redes de Places365, de 365 hasta 8 [Figura A.3]. Finalmente, se integra las 4 redes (con salida de tamaño 8 cada una) en una única red, de salida con tamaño 8 [Figura A.7].

De esta forma, tanto para el entrenamiento como para la validación, se alimentará a cada una de las redes con el dataset de su escala respectiva, obteniéndose a la salida final un único vector o tensor con las probabilidades de cada una de las 8 clases.

Capítulo 5

Pruebas y resultados finales

5.1. Introducción

En este capítulo se estudiarán los distintos resultados obtenidos tras la elección de los parámetros de creación de las diferentes escalas y el número de muestras para el reconocimiento de escenas.

5.2. Primeras pruebas

Uno de los aspectos más importantes de este trabajo es la multiescala en imágenes para el reconocimiento de redes convolucionales. La elección de 4 escalas para ello, trae consigo la óptima selección de éstas. Como consecuencia de ello, se debe primero probar la utilización de diferentes escalas y comprobar su funcionamiento y nivel de optimización. Pare ello se llevará un pequeño estudio, modificando los parámetros de cada escala hasta que se obtenga de la red una pérdida aproximada de 0.001 y observando el número de epochs (ejecuciones) que tarda cada una en llegar a ese valor. Una tasa de pérdida como la que se busca, es un indicativo de un buen aprendizaje de la red.

5.2.1. Pruebas con GS

Para las imágenes de mayor escala se establece que el dataset GS, esté conformado por las imágenes originales reescaladas al tamaño de entrada en la red (224x224). Por tanto, para esta escala se determina una escala fija que no estará bajo estudio.

5.2.2. Pruebas con GI

En cuanto a la escala GI, se ha probado con diferentes medidas de reescalado y un diferente número de muestras (pequeño). Se han obtenido los siguientes valores en el entrenamiento con Places365.

Tamaño/Nº muestras	2	3	4	5
600*600	18	17	18	19
500*500	18	17	18	20
400*400	17	16	19	19
300*300	16	15	18	20

Figura 5.1: Valores de entrenamiento para escala GI

Tras observar que los valores no sufren un cambio significativo entre ellos, se ha establecido que este dataset esté formado por imágenes reescaladas a 300x300 y tres muestras aleatorias. Se toma esta decisión, debido a que: a mayor número de imágenes que se pasan a la red, ésta aprende igual que con un número menor.

5.2.3. Pruebas con FS

Se continúa con el estudio de la primera de las escalas de menor dimensión, la escala FS. Se ha probado con ciertos valores de reescalado (alto) y diferentes muestras en función de éste. Se observa más abajo los valores arrojados durante el entrenamiento con los diferentes parámetros.

Tamaño/Nº muestras	10	15	20	25	30
85%	23	22	24	24	25
80%	22	20	23	25	24
75%	23	21	23	23	24
70%	22	21	22	22	23

Figura 5.2: Valores de entrenamiento para escala FS

Se puede observar una uniformidad en los valores del entrenamiento,con lo cual, se ha establecido que este dataset esté formado por imágenes reescaladas a un 80 % del tamaño original y 15 muestras aleatorias de la misma. Se toma esta decisión debido a que, aunque a mayor número de imágenes que se pasan a la red, esta aprende igual, las características de este dataset son diferentes que para escalas mayores.

5.2.4. Pruebas con FI

Por último, la escala más fina (FI) debe estar formada por un número (grande) de imágenes de muestra de la imagen original. A continuación se detalla la información de diferentes valores de muestras en el entranamiento para Imagenet.

Nº muestras	
30	23
35	22
40	23
45	22
50	22

Figura 5.3: Valores de entrenamiento para escala FI

Al igual que en la escala anterior, no hay diferencias sustanciales, por lo que se ha establecido tomar 30 muestras de cada imagen del dataset original.

5.3. Entrenamiento

Una vez elegida la escala y número de muestras de cada una, se procede al entrenamiento de la red integrada con sus respectivos datasets. Como se indicó en el Capítulo 3.6 Fine Tuning, al haberse añadido capas fully-connected nuevas a las redes, éstas actualmente se encuentran vacías, es decir no tienen conocimiento alguno. Debido a esto se procede a realizar un pequeño entrenamiento de estas nuevas capas, que consistirá solo de un (dos para las escalas pequeñas) epoch, para cada una de la redes internas. Haciendo uso de la opción que brinda Pytorch, se desactivan todas las capas de la red, menos las capas nuevas (fully-connected), como se observa en la Figura A.8. Esto se implementa para aportar algo de conocimiento a estas nuevas capas, para que a la hora de realizar el entrenamiento completo, se parta de una pérdida menor. A partir de ahora se mostrará la información de entrenamiento (epoch, tamaño de batch y pérdida) de la red con todas sus capas activas, como muestra la Figura A.9.

5.3.1. Entrenamiento para GS

Se procede con el entrenamiento de la red (pre-entrenada con el dataset de Places365) con un dataset, el de GS, de imágenes originales reescaladas a 224x224. Al tratarse de imágenes de gran escala, el aprendizaje de la red es bastante rápido. En escasos epochs se consigue obtener una pérdida deseable de 0.001.

```
Starting epoch 1 / 10
[1,    36] loss: 0.273
Starting epoch 2 / 10
[2,    36] loss: 0.118
Starting epoch 3 / 10
[3,    36] loss: 0.070
Starting epoch 4 / 10
[4,    36] loss: 0.050
Starting epoch 5 / 10
[5,    36] loss: 0.037
Starting epoch 6 / 10
[6,    36] loss: 0.019
Starting epoch 7 / 10
[7,    36] loss: 0.008
Starting epoch 8 / 10
[8,    36] loss: 0.009
Starting epoch 9 / 10
[9,    36] loss: 0.003
Starting epoch 10 / 10
[10,    36] loss: 0.001
```

Figura 5.4: Valores de entrenamiento para escala GS

5.3.2. Entrenamiento para GI

Se lleva a cabo el entrenamiento de la red (pre-entrenada con el dataset de Places365) con un dataset, el de GI, con diferentes imágenes de muestra (3), tras haber reescalado a un tamaño de 300x300. Ahora, en cambio, esta red tarda un poco más en aprender (obtener pérdida deseada de 0.001), ya que hay pequeñas diferencias entre

las imágenes.

```
Starting epoch 1
[1,  107] loss: 0.856
Starting epoch 2
[2,  107] loss: 0.521
Starting epoch 3
[3,  107] loss: 0.435
Starting epoch 4
[4,  107] loss: 0.296
Starting epoch 5
[5,  107] loss: 0.174
Starting epoch 6
[6,  107] loss: 0.210
Starting epoch 7
[7,  107] loss: 0.162
Starting epoch 8
[8,  107] loss: 0.109
Starting epoch 9
[9,  107] loss: 0.009
Starting epoch 10
[10,  107] loss: 0.008
Starting epoch 11
[11,  107] loss: 0.006
Starting epoch 12
[12,  107] loss: 0.006
Starting epoch 13
[13,  107] loss: 0.003
Starting epoch 14
[14,  107] loss: 0.003
Starting epoch 15
[15,  107] loss: 0.001
```

Figura 5.5: Valores de entrenamiento para escala GI

5.3.3. Entrenamiento para FS

Se continúa con el entrenamiento de la red (pre-entrenada con el dataset de ImageNet) con un dataset, el de FS, con diferentes muestras de una imagen, tras haber reescalado a un tamaño del 80 % del original. Como se puede observar, esta red requiere un tiempo considerable para aprender. Esto es debido a las características de este dataset, compuesto por imágenes totalmente diferentes unas de otras, incluso pertenecientes a una misma imagen original. La pérdida en este caso, varía de una forma menor en cada iteración. Debido al gran volumen de imágenes, el número de epochs que tarda en obtener una pérdida de 0.001 es bastante mayor.

```
Starting epoch 1
[1,   639] loss: 0.936
Starting epoch 2
[2,   639] loss: 0.833
Starting epoch 3
[3,   639] loss: 0.581
Starting epoch 4
[4,   639] loss: 0.340
Starting epoch 5
[5,   639] loss: 0.149
Starting epoch 6
[6,   639] loss: 0.094
Starting epoch 7
[7,   639] loss: 0.078
Starting epoch 8
[8,   639] loss: 0.064
Starting epoch 9
[9,   639] loss: 0.056
Starting epoch 10
[10,   639] loss: 0.045
Starting epoch 11
[11,   639] loss: 0.035
Starting epoch 12
[12,   639] loss: 0.012
Starting epoch 13
[13,   639] loss: 0.009
Starting epoch 14
[14,   639] loss: 0.008
Starting epoch 15
[15,   639] loss: 0.005
Starting epoch 16
[16,   639] loss: 0.003
Starting epoch 17
[17,   639] loss: 0.005
Starting epoch 18
[18,   639] loss: 0.004
Starting epoch 19
[19,   639] loss: 0.003
Starting epoch 20
[20,   639] loss: 0.001
```

Figura 5.6: Valores de entrenamiento para escala FS

5.3.4. Entrenamiento para FI

Por último, se entrena a la red (pre-entrenada con el dataset de ImageNet) con un dataset, el de FI, con diferentes muestras de una misma imagen. El tiempo continua aumentando, al igual que para la escala FS, debido a las características propias del dataset de esta escala, requiere muchos epochs para alcanzar la tasa de pérdida deseada.

```
Starting epoch 1
[1,  1242] loss: 0.697
Starting epoch 2       Starting epoch 13
[2,  1242] loss: 0.421 [13,  1242] loss: 0.033
Starting epoch 3       Starting epoch 14
[3,  1242] loss: 0.293 [14,  1242] loss: 0.032
Starting epoch 4       Starting epoch 15
[4,  1242] loss: 0.216 [15,  1242] loss: 0.029
Starting epoch 5       Starting epoch 16
[5,  1242] loss: 0.174 [16,  1242] loss: 0.017
Starting epoch 6       Starting epoch 17
[6,  1242] loss: 0.139 [17,  1242] loss: 0.019
Starting epoch 7       Starting epoch 18
[7,  1242] loss: 0.122 [18,  1242] loss: 0.011
Starting epoch 8       Starting epoch 19
[8,  1242] loss: 0.104 [19,  1242] loss: 0.007
Starting epoch 9       Starting epoch 20
[9,  1242] loss: 0.091 [20,  1242] loss: 0.009
Starting epoch 10      Starting epoch 21
[10,  1242] loss: 0.082[21,  1242] loss: 0.005
Starting epoch 11      Starting epoch 22
[11,  1242] loss: 0.041[22,  1242] loss: 0.003
Starting epoch 12      Starting epoch 23
[12,  1242] loss: 0.035[23,  1242] loss: 0.001
```

Figura 5.7: Valores de entrenamiento para escala FI

Nota: para el entrenamiento de cada escala solo se encuentra activa la red para la cual está enfocada cada escala .

5.4. Validación

Tras haber entrenado la red integrada, es hora de validar si su aprendizaje/entrenamiento ha sido correcto o no, y hasta que punto es óptimo. Al igual que en el entrenamiento, se le pasa a las diferentes redes imágenes de validación de sus respectivos datasets. Éstas imágenes de testeo han de ser siempre diferentes a las usadas en el entrenamiento. La creación de estos datasets ha tenido lugar durante la creación de los mismos para el entrenamiento.

5.4.1. Validación sin integración

Antes de proceder con la validación de la red intregrada, se lleva a cabo la validación de las 4 redes que conforman la red integrada, de forma independiente. Cada uno

de los 4 vectores de salida de cada red, posee tamaño 8. Para afrontar la validación, se ha evaluado el porcentaje de acierto de cada red sobre cada clase, y por último; un promedio de acierto de las 8 clases sobre la misma red. Tras esta validación se obtienen los siguientes resultados:

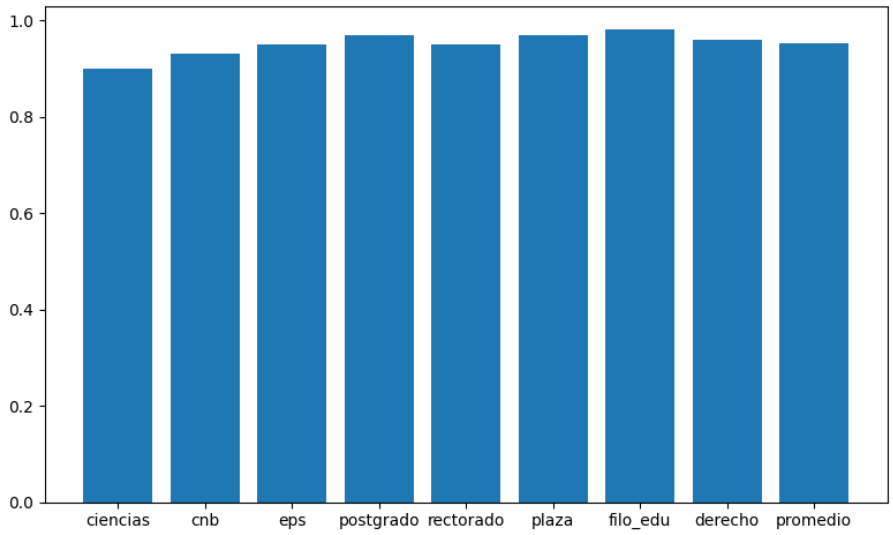

Figura 5.8: Porcentaje de acierto para la escala GS y su promedio total

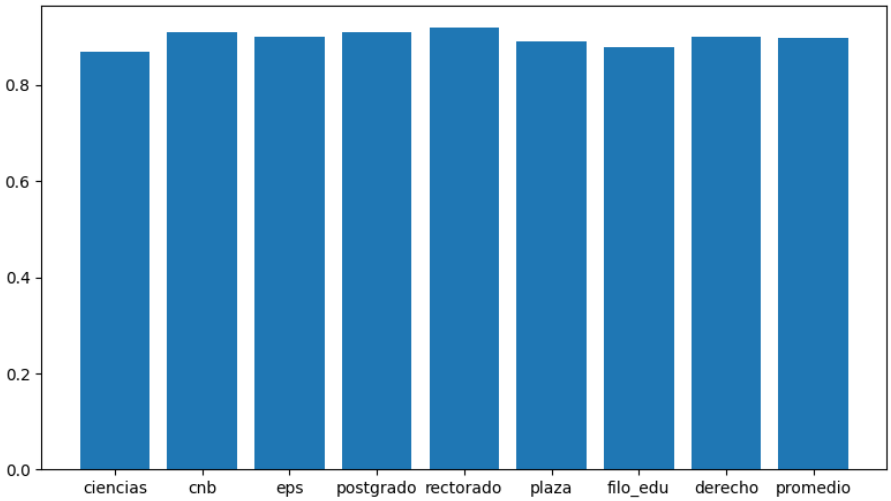

Figura 5.9: Porcentaje de acierto para la escala GI y su promedio total

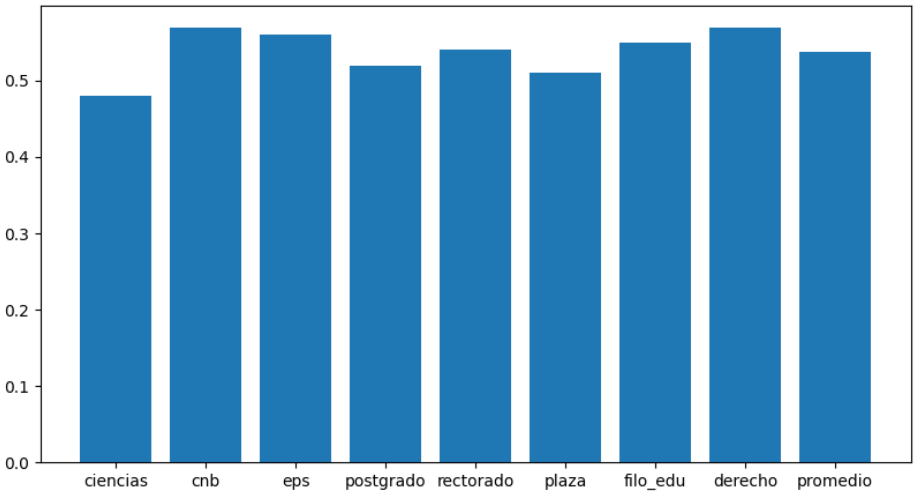

Figura 5.10: Porcentaje de acierto para la escala FS y su promedio total

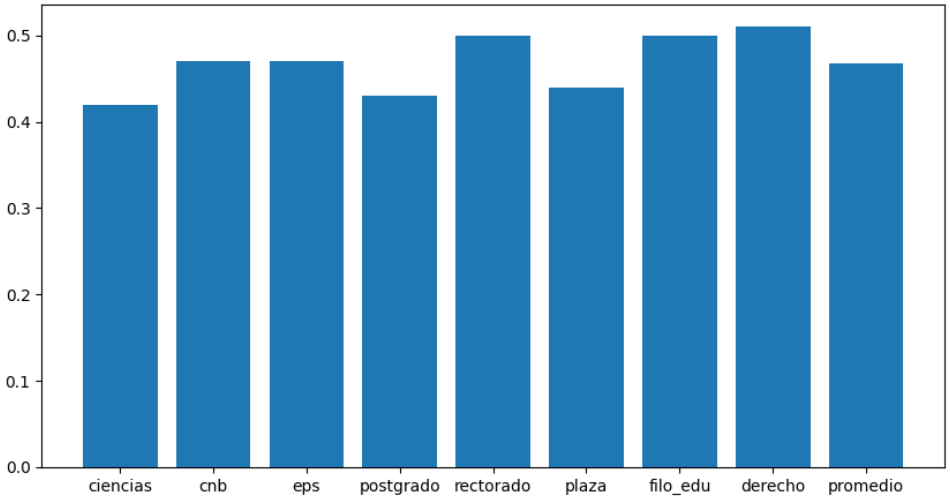

Figura 5.11: Porcentaje de acierto para la escala FI y su promedio total

Se estiman los siguientes intervalos para los valores de validación, todos ellos en porcentaje:

- [50,60): Resultado obtenido es malo.
- [60,75): Resultado obtenido es medio.
- [75,90): Resultado obtenido es bueno.
- [90,97): Resultado obtenido es muy bueno.
- [97,100): Resultado obtenido es excelente.

Con ello se observa la gran especificidad de las imágenes de gran escala (tanto GS, como GI) de las redes entrenadas con Places365, consecuencia de la menor variabilidad de este tipo de imágenes, que permite obtener unos valores muy altos de acierto. Los resultados se contraponen a los obtenidos en imágenes de pequeña escala (FS y FI), que permiten un mayor nivel de acierto, sí dentro de ellas se encuentran objetos particulares que sean característicos de una escena, el caso contrario hará que se acierte en menor medida. Por lo tanto, una vez visto como la escala es determinante a la hora de obtener un mayor acierto, se continua con la validación de toda la red integrada.

5.4.2. Validación de la red integrada

Tras haber realizado la validación por separado, se procede al testeo de la red integrada. En este caso se pasa a la red, cuatro imágenes, una por cada escala y se obtiene a la salida un único vector (tensor). De esta forma se obtiene el siguiente resultado:

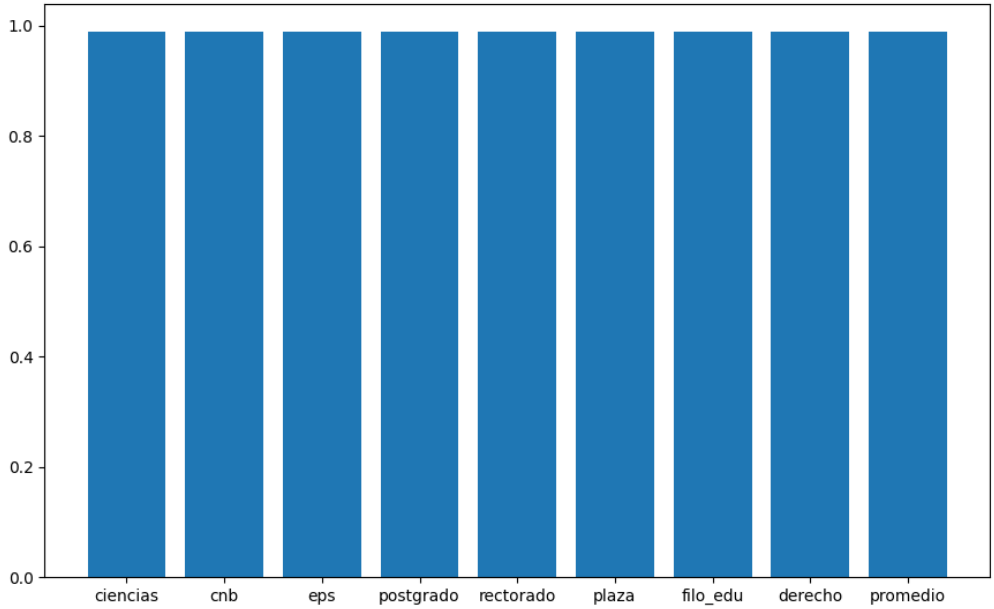

Figura 5.12: Porcentaje de acierto para la red integrada y su promedio total

Como se puede observar, el porcentaje es incluso mayor que el obtenido con las redes por separado. Lo que ocurre es fruto de la integración de toda la red, ahora se dispone de todo el conocimiento obtenido por ésta durante el entrenamiento, y se logra que el acierto conste de un nivel casi perfecto (el promedio de acierto de todas las clases es del 98.88 %). Esto es debido a la complementación del conocimiento de cada una de las redes, a partir del entrenamiento en cada escala respectiva, lo cual conlleva que; aunque una de las redes internas no acierte la clase, el resto de redes aporta ese conocimiento complementario; que consigue elevar eficazmente el porcentaje de acierto.

A continuación, se ha realizado un estudio más profundo del resultado del testeo

enfocado a cada una de las clases del dataset.

5.4.3. EPS

En el caso concreto de la imágenes pertenecientes a la clase EPS, se obtiene el siguiente nivel de acierto:

- Porcentaje de acierto de EPS: 100 %

Como se observa en el gráfico inferior, el elevado nivel de acierto se debe a las características particulares de la EPS: edificio de ladrillo con un color particular junto con árboles y césped verdes.

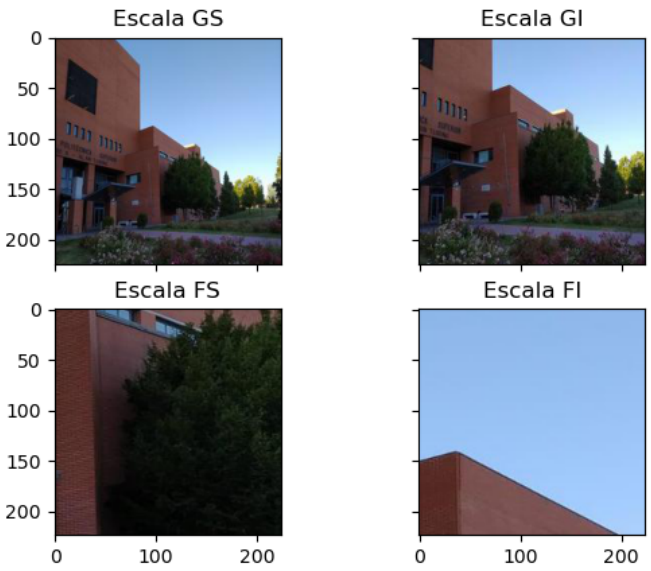

Figura 5.13: Imagen de muestra de EPS y sus escalas

5.4.4. Postgrado

Para las imágenes pertenecientes a la clase Postgrado, se obtiene el siguiente nivel de acierto:

- Porcentaje de acierto de Postgrado: 100 %

Como se puede observar en la gráfica inferior, el absoluto nivel de acierto es a causa de las características distintivas del edificio de Postgrado de la Universidad:

edificio blanco/gris con una escalera particular rodeada de árboles. Todo ello hace que el reconocimiento de esta escena sea medianamente fácil para el reconocedor.

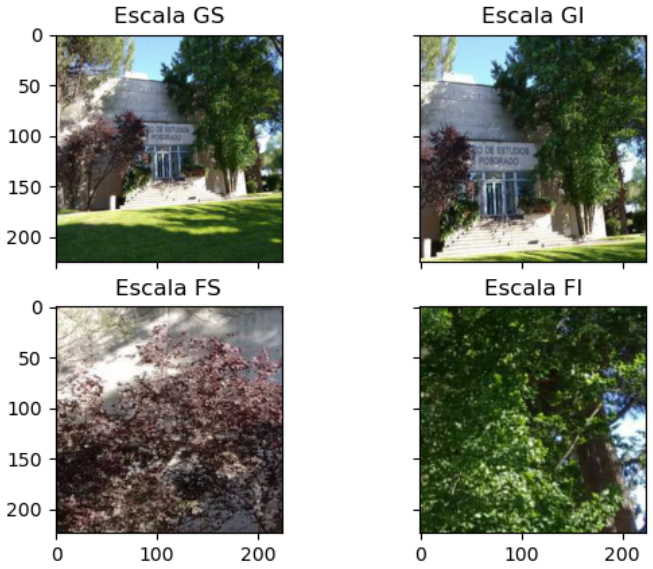

Figura 5.14: Imagen de muestra de Postgrado y sus escalas

5.4.5. CNB

Para las imágenes pertenecientes a la clase CNB (Centro Nacional de Biotcnología), se obtiene el siguiente nivel de acierto:

- Porcentaje de acierto de CNB: 100 %

Siendo este un edificio tan distintivo por sus líneas verticales blancas y verdes, junto con las ventanas redondas, es completamente normal obtener un nivel de acierto

en validación.

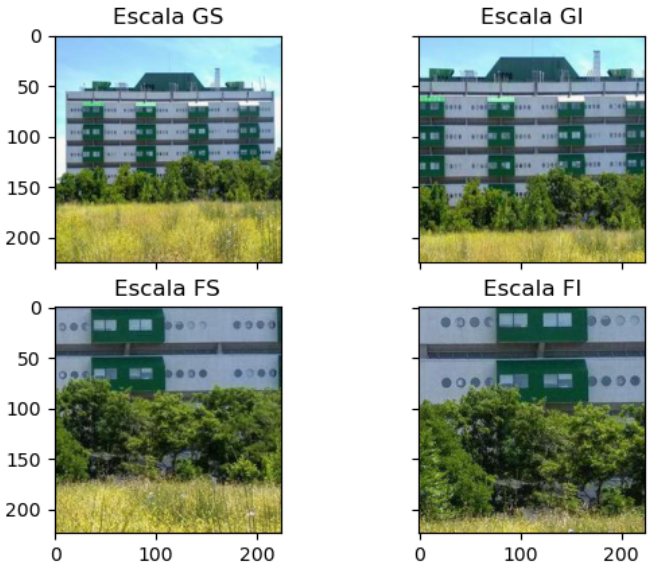

Figura 5.15: Imagen de muestra de CNB y sus escalas

5.4.6. Filosofía y Educación

A la hora de validar las imágenes pertenecientes a la clase Filosofía y Educación, se obtiene el siguiente nivel de acierto:

- Porcentaje de acierto de Filo_Edu: 99 %

Como se observa en la validación de la facultad de Ciencias más adelante, la facultad de Filosofía y Educación no tiene unas características tan distiguibles, y además posee algunas en común con ella debido a su estilo arquitectónico. De esta

forma seconsigue un valor que no es perfecto como en las anteriores clases.

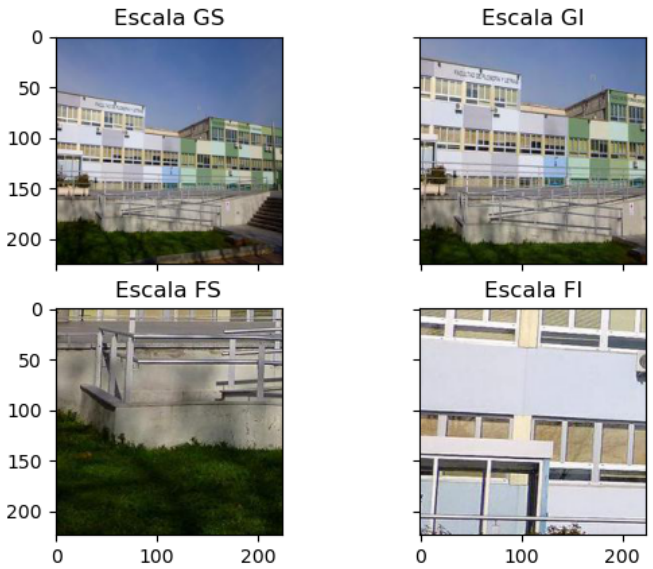

Figura 5.16: Imagen de muestra de Filosofía y Educación y sus escalas

5.4.7. Plaza mayor

Continuando con las imágenes pertenecientes a la clase Plaza Mayor, se obtiene el siguiente nivel de acierto:

- Porcentaje de acierto de Plaza: 98 %

Se consigue para esta clase un acierto muy alto, casi perfecto. Otra vez más, es debido a la particularidad de las carecterísticas de este lugar: edificio de color claro

muy parecido al color del cielo, lo cual lo hace bastante distinguible del resto.

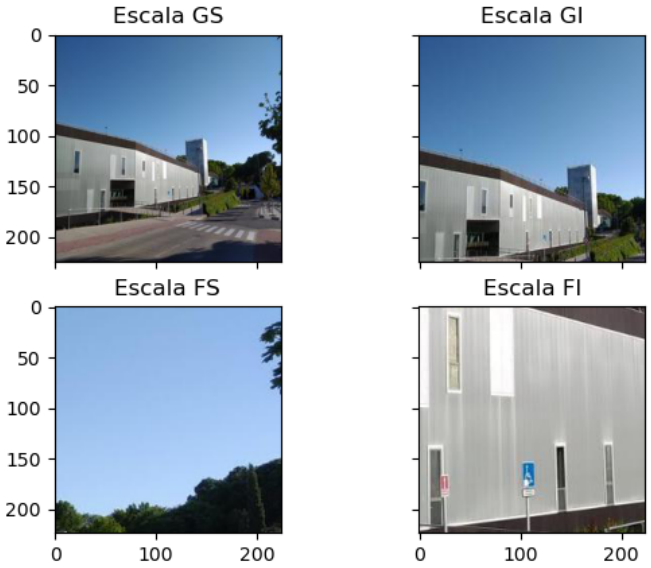

Figura 5.17: Imagen de muestra de Plaza Mayor y sus escalas

5.4.8. Ciencias

Se continua con la validación las imágenes pertenecientes a la clase Ciencias, se obtiene el siguiente nivel de acierto:

- Porcentaje de acierto de Ciencias: 97 %

Como se ha visto con la clase de Filosofía y Educación, ambas contienen elementos arquitectónicos en común, pero a su vez aspectos muy particulares como son la escalera, el color azul y las dos columas delanteras. Todo esto conlleva a conseguir un nivel de acierto más bajo que el de la media de todo el dataset.

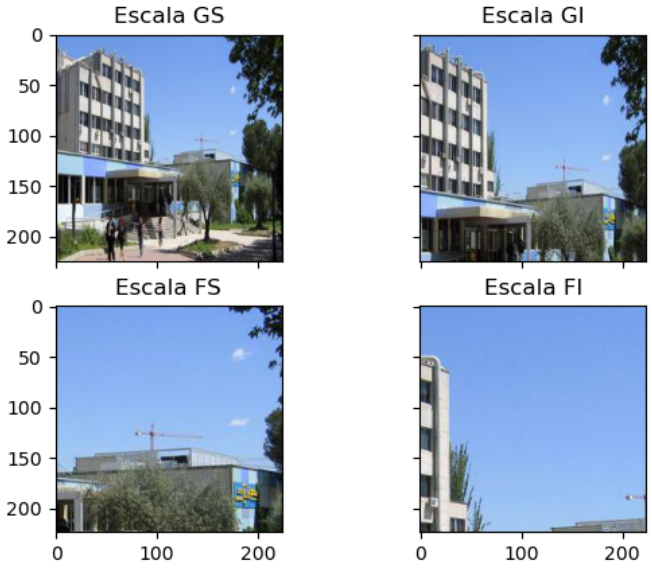

Figura 5.18: Imagen de muestra de Ciencias y sus escalas

5.4.9. Rectorado

En cuanto a las imágenes pertenecientes a la clase Rectorado, se obtiene el siguiente nivel de acierto:

- Porcentaje de acierto de Rectorado: 100 %

Como se puede observar en la gráfica, el Rectorado es un edificio inconfundible, como se puede admirar, consta de un color oscuro en la fachada. Ésta a su vez está formada por pequeños azulejos de diferente color. También cabe destacar la forma de

las ventanas. Todo lo anterior, confirma el perfecto nivel de acierto obtenido.

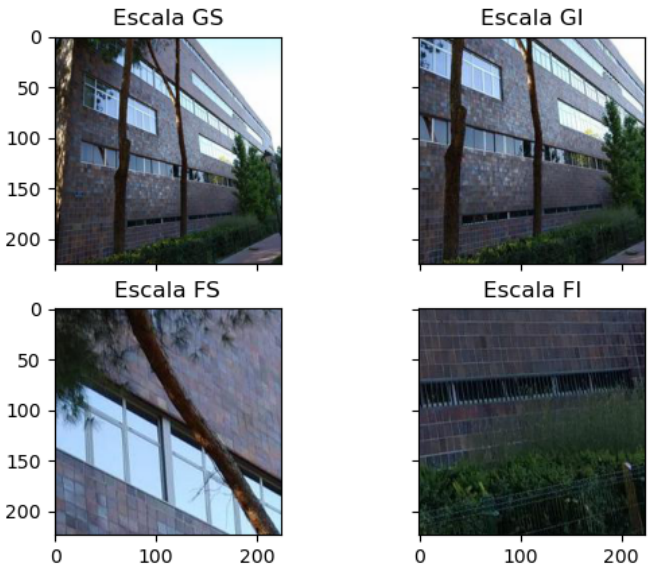

Figura 5.19: Imagen de muestra de Rectorado y sus escalas

5.4.10. Derecho

Por último, para la imágenes pertenecientes a la clase Derecho, se obtiene el siguiente nivel de acierto:

- Porcentaje de acierto de Derecho: 100 %

Como se observa en la gráfica de más abajo, el perfecto nivel de acierto se debe a las características particulares de la facultad de Derecho: edificio de ladrillo con un

color particular junto a una escalera de color claro y rodeado de árboles.

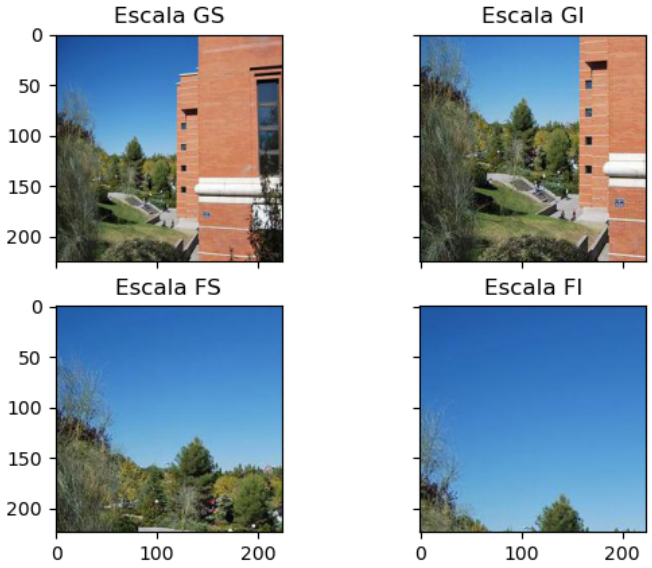

Figura 5.20: Imagen de muestra de Derecho y sus escalas

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusiones

En este trabajo se ha llevado a cabo el desarrollo e implementación de diferentes redes neuronales destinadas a reconocer diferentes lugares de la Universidad, con la particularidad de crear diferentes dataset propios de determinadas escalas y ver su efecto e influencia a la hora de reconocer estas escenas. Se ha logrado analizar los distintos comportamientos que tienen las redes en función del dataset con el que han sido entrenadas, tanto con Imagenet para escalas pequeñas, como Places365 para escalas grandes. Partiendo de redes ya creadas, se han realizado las transformaciones necesarias para lograr realizar el estudio del principal objetivo del trabajo. He profundizado en conceptos e ideas que eran desconocidas para mí antes de abordar este trabajo, siendo ahora conocedor de las diferentes herramientas y características relacionadas con el entrenamiento y validación de redes neuronales convolucionales, siendo capaz de distinguir la utilidad de cada una de las capas que forman la arquitectura de éstas. La curva de aprendizaje la definiría como creciente y secuencial, con el paso del tiempo durante el desarrollo, los conceptos iban asentándose hasta llegar a ser capaz de realizar un análisis profundo de lo observado durante el trabajo.

De la misma manera, este Trabajo de Fin de Grado ha sido de gran utilidad para aprender e introducirme en el campo del Deep Learning, así como poner en práctica lenguajes de programación nuevos para mí, como Python (y el gran número de librerías de este campo, entre ellas destacaría Pytorch) y entornos de desarrollo como Pycharm. Así mismo, me ha permitido profundizar en conceptos de este campo como las redes neuronales y ser capaz de enfrentarme de esta manera a un caso de uso y estudio de esta rama. He ahondado en este conocimiento mediante la búsqueda en webs, participación en foros y lectura de documentación online.

6.2. Trabajo futuro

Una vez finalizado este trabajo y teniendo un ligero conocimiento del área, se pueden establecer líneas de desarrollo del mismo, mediante distintas posibilidades que se podrían llevar a la práctica realizando nuevas investigaciones. A continuación se enuncian posibles o futuras mejoras con las que optimizar elementos de este proyecto:

- Llevar a cabo el estudio mediante diferentes redes neuronales destinadas al reconocimiento de imágenes.
- Aumentar el dataset propio de lugares de la universidad, o añadir distintas imágenes de los edificios ya existentes.
- Una vez aumentada la base de datos, tratar de añadir escenas interiores de los distintos edificios y comparar resultados entre ambos.
- Realizar este mismo trabajo con diferentes frameworks de Deep Learning (Keras, TensorFlow, ...) y realizar un estudio para comprobar distintos resultados.
- Tratar de desarrollar una aplicación de la Universidad Autónoma basada en este proyecto que sirva de guía dentro del campus.

Bibliografía

[1] L. Herranz, S. Jiang, and X. Li, "Scene recognition with cnns: objects, scales and dataset bias," *CoRR*, vol. abs/1801.06867, 2018.

[2] A. Hidaka and T. Kurita, "Consecutive dimensionality reduction by canonical correlation analysis for visualization of convolutional neural networks," vol. 2017, pp. 160–167, 12 2017.

[3] J. Peng, S. Kang, Z. Ning, H. Deng, J. Shen, Y. Xu, J. Zhang, W. Zhao, X. Li, W. Gong, J. Huang, and L. Liu, "Residual convolutional neural network for predicting response of transarterial chemoembolization in hepatocellular carcinoma from ct imaging," *European Radiology*, vol. 30, pp. 1–12, 07 2019.

[4] M. Garland, S. Le Grand, J. Nickolls, J. Anderson, J. Hardwick, S. Morton, E. Phillips, Y. Zhang, and V. Volkov, "Parallel computing experiences with cuda," *IEEE Micro*, vol. 28, no. 4, pp. 13–27, 2008.

[5] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into Imaging*, vol. 9, pp. 611–629, Aug 2018.

[6] M. Liang and X. Hu, "Recurrent convolutional neural network for object recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[7] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," *CoRR*, vol. abs/1912.01703, 2019.

[8] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, (Savannah, GA), pp. 265–283, USENIX Association, Nov. 2016.

[9] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *CoRR*, vol. abs/1408.5093, 2014.

[10] F. e. a. Chollet, "Keras," 2015.

[11] S. Targ, D. Almeida, and K. Lyman, "Resnet in resnet: Generalizing residual architectures," *CoRR*, vol. abs/1603.08029, 2016.

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[13] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[14] E. Cetinic, T. Lipic, and S. Grgic, "Fine-tuning convolutional neural networks for fine art classification," *Expert Systems with Applications*, vol. 114, pp. 107–118, 2018.

Apéndice A

En esta sección se mostrará el código implementado para el desarrollo que se ha realizado en este proyecto.

```
)
(avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
(fc): Linear(in_features=2048, out_features=365, bias=True)
```

Figura A.1: Penúltima y última capa de ResNet50 para Places365

```
)
(avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
(fc): Linear(in_features=2048, out_features=1000, bias=True)
```

Figura A.2: Penúltima y última capa de ResNet50 para Imagenet

```
model.fc = nn.Sequential(
    nn.Linear(model.fc.in_features, 365),
    nn.Linear(365, 128),
    nn.Linear(128, 64),
    nn.Linear(64, num_classes)
)
```

Figura A.3: Modificación última capa para Places365

```
model.fc = nn.Sequential(
    nn.Linear(model.fc.in_features, 1000),
    nn.Linear(1000, 512),
    nn.Linear(512, 128),
    nn.Linear(128, 64),
    nn.Linear(64, num_classes)
)
```

Figura A.4: Modificación última capa para Imagenet

```
(avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
(fc): Sequential(
  (0): Linear(in_features=2048, out_features=365, bias=True)
  (1): Linear(in_features=365, out_features=128, bias=True)
  (2): Linear(in_features=128, out_features=64, bias=True)
  (3): Linear(in_features=64, out_features=8, bias=True)
)
```

Figura A.5: Penúltima y última capa de ResNet50 para Places365 tras modificación

```
(avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
(fc): Sequential(
  (0): Linear(in_features=2048, out_features=1000, bias=True)
  (1): Linear(in_features=1000, out_features=512, bias=True)
  (2): Linear(in_features=512, out_features=128, bias=True)
  (3): Linear(in_features=128, out_features=64, bias=True)
  (4): Linear(in_features=64, out_features=8, bias=True)
)
```

Figura A.6: Penúltima y última capa de ResNet50 para Imagenet tras modificación

```python
class Integracion(nn.Module):
    def __init__(self, modelgs, modelgi, modelfs, modelfi):
        super(Integracion, self).__init__()
        self.modelfi = modelfi
        self.modelfs = modelfs
        self.modelgi = modelgi
        self.modelgs = modelgs
        self.classifier = nn.Linear(4*8, 8)

    def forward(self, x1, x2, x3, x4):
        x1 = self.modelgs(x1)
        x2 = self.modelgi(x2)
        x3 = self.modelfs(x3)
        x4 = self.modelfi(x4)
        x = torch.cat((x1, x2, x3, x4), dim=1)
        x = self.classifier(F.relu(x))
        return x
```

Figura A.7: Integración de redes en una sola

```python
for param in model.parameters():
    param.requires_grad = False
for param in model.fc.parameters():
    param.requires_grad = True
```

Figura A.8: Activación única de capa fully-connected

```python
for param in model.parameters():
    param.requires_grad = True
for param in model.fc.parameters():
    param.requires_grad = True
```

Figura A.9: Activación de toda la red

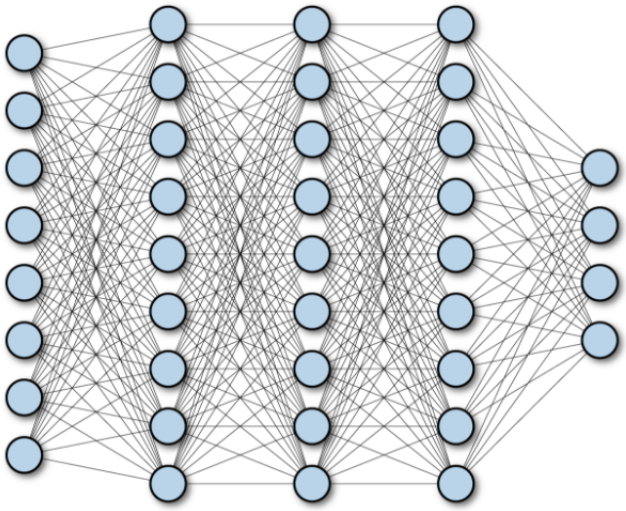

Figura A.10: Capa fully-connected

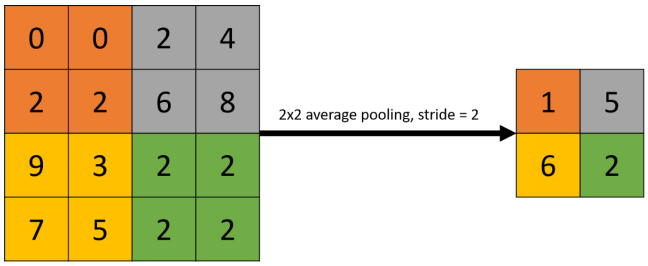

Figura A.11: Capa Average Pool